High Performance Computational Dynamics at UW-Madison

Associate Prof. Dan Negrut
NVIDIA CUDA Fellow
&
PhD Candidate Toby Heyn
Simulation-Based Engineering Lab
Department of Mechanical Engineering
University of Wisconsin – Madison
Acknowledgements

- **People:**
  - Alessandro Tasora – University of Parma, Italy
  - Mihai Anitescu – Argonne National Lab
  - Lab Students:
    - Aaron Bartholomew
    - Makarand Datar
    - Toby Heyn
    - Naresh Khude
    - Justin Madsen
    - Hammad Mazhar
    - Dan Melanz
    - Spencer O'Rourke
    - Arman Pazouki
    - Andrew Seidl

- **Financial support**
  - National Science Foundation, Young Investigator Career Award
  - Army Research Office (ARO)
  - US Army TARDEC
  - FunctionBay, S. Korea
  - NVIDIA
  - Caterpillar
  - NASA (Jet Propulsion Lab)
Talk Overview

- Intro: lab’s vision/strategy/priorities in Computational Dynamics
- The role of quadratic programming in Computational Dynamics
- The need for high performance computing in Computational Dynamics
  - Heterogeneous Computing Template (HCT)
- Conclusions and directions of future work
Vision/Strategy/Focus

- **Vision**
  - Share and discover new ways in which modeling, simulation, and visualization can foster innovation and scientific discovery

- **Strategy**
  - Leverage emerging hardware architectures and develop/use novel numerical algorithms to advance the state of the art in physics-based simulation

- **Focus Area**
  - Engineering, focusing on multi-body dynamic systems
Computational Dynamics Example Application
Classical Computational Dynamics

\[
\dot{q} = L(q) v \\
M(q) \ddot{v} = f(t, q, v) - g_q^T(q, t) \lambda \\
g(q, t) = 0
\]
\[ h = 0.0001 \,[s] \]
\[ g = -9.80665 \,\frac{m}{s^2} \]
20k spheres
\[ r = 3.5 \,mm \]
\[ \mu = 0.46 \]

\[ \omega = \pi \,\frac{\text{rad}}{\text{sec}} \]

*Anchor width = 5 \, [cm]*
Frictional Contact Simulation
[Commercial Solution]

- Model Parameters:
  - Spheres: 60 mm diameter and mass 0.882 kg
  - Forces: smoothing with stiffness of 1E5, force exponent of 2.2, damping coefficient of 10.0, and a penetration depth of 0.1
  - Simulation length: 3 seconds

![Simulation Image]

**CPU time v. Number of Spheres in ADAMS**

\[ y = 0.8385x^2 - 7.2607x + 16.154 \]

\[ R^2 = 0.9985 \]
An Engineering Application...

- How is the Rover moving along on a slope with granular material?
- What wheel geometry is more effective?
Handling Frictional Contact Phenomena

- Two broadly used approaches for handling frictional contact:
  - Soft-body approaches
    - Called DEM solutions
  - Hard-body approaches
    - Called DVI solutions

- This talk focuses on the DVI approach → draws heavily on quadratic programming techniques…
The Relevant Problem $\Rightarrow$

Two Common Solutions $\Rightarrow$

Solution Draws on $\Rightarrow$
The DVI Framework...

- The hard-body approach: two rigid bodies in contact shall move so that their boundaries are not overlapping.

- There is a complementarity condition that captures this requirement:

\[ 0 \leq \Phi(q, t) \perp \gamma_n \geq 0 \]
There is also friction between bodies (acts in the tangent plane):

$$F_f = \gamma_u t_u + \gamma_w t_w$$

Coulomb friction model states that the following conditions hold

$$0 \leq \mu^2 \gamma_n^2 - (\gamma_u^2 + \gamma_w^2) \quad \perp \quad \|v_T\| \geq 0$$

$$\langle v_T, F_f \rangle = -\|v_T\| \cdot \|F_f\|$$
The DVI Framework...

(Cntd.)

• An equivalent way of stating the Coulomb friction model is

\[(\gamma_u^*, \gamma_w^*) = \arg \min_{\gamma_u^2 + \gamma_w^2 - \mu^2 \gamma_n^2 \leq 0} \left[ v^T (\gamma_u t_u + \gamma_w t_w) \right]\]

• Recall that

\[F_f = \gamma_u^* t_u + \gamma_w^* t_w\]

• In other words, the friction force should be such that the relative motion between the two bodies maximizes the amount of power dissipated
Many-Body Dynamics

\[ \dot{q} = L(q)v \]

\[ M(q)\ddot{v} = f(t, q, v) - g_q^T(q, t)\lambda + \sum_{i=1}^{N_c} \left( \gamma_n^i D_{n,i}^T + \gamma_u^i D_{u,i}^T + \gamma_w^i D_{w,i}^T \right) \]

\[ g(q, t) = 0 \]

Contact Impulse, for Contact “i”

\[ 0 \leq \Phi^i(q, t) \quad \perp \quad \gamma_i^i \geq 0 \quad i = 1, 2, \ldots, N_c \]

Friction Impulse Components, for Contact “i”

Gap Function, for Contact “i”

Friction Dissipation Energy

Total Number of Contacts
Mixing 50,000 M&Ms on the GPU
Traditional Discretization Scheme

\[
\begin{align*}
q^{(l+1)} &= q^{(l)} + hL(q^{(l)})v^{(l+1)} \\
M(v^{(l+1)} - v^l) &= hf(t^{(l)}, q^{(l)}, v^{(l)}) + \sum_{i\in\mathcal{A}(q^{(l)}, \delta)} (\gamma_{i,n} D_{i,n} + \gamma_{i,u} D_{i,u} + \gamma_{i,w} D_{i,w})
\end{align*}
\]

\[
i \in \mathcal{A}(q^{(l)}, \delta): \quad 0 \leq \frac{1}{h} \Phi_i(q^{(l)}) + D_{i,n}^T v^{(l+1)} \perp \gamma_{i,n} \geq 0,
\]

\[
(\gamma_{i,u}, \gamma_{i,w}) = \arg\min_{\mu_i \gamma_{i,n} \geq \sqrt{\gamma_{i,u}^2 + \gamma_{i,w}^2}} v^T (\gamma_{i,u} D_{i,u} + \gamma_{i,w} D_{i,w}).
\]

(Stewart, 1998)
Relaxed Discretization Scheme Used

\[ q^{(l+1)} = q^{(l)} + hL(q^{(l)})v^{(l+1)} \]

\[
\mathbf{M}(v^{(l+1)} - v^l) = hf(t^{(l)}, q^{(l)}, v^{(l)}) + \sum_{i \in A(q^{(l)})} \left( \gamma_{i,n} D_{i,n} + \gamma_{i,u} D_{i,u} + \gamma_{i,w} D_{i,w} \right)
\]

\[ i \in A(q^{(l)}, \delta) : \quad 0 \leq \frac{1}{h} \Phi_i(q^{(l)}) + D^T_{i,n} v^{(l+1)} - \mu^i \sqrt{(v^T D_{i,u})^2 + v^T D_{i,w})^2} \quad \gamma^i \geq 0, \]

\[ (\gamma_{i,u}, \gamma_{i,w}) = \arg\min_{\mu_i \gamma_{i,n} \geq \sqrt{\gamma_{i,u}^2 + \gamma_{i,w}^2}} v^T (\gamma_{i,u} D_{i,u} + \gamma_{i,w} D_{i,w}). \]

(Anitescu & Tasora, 2008)
The Cone Complementarity Problem (CCP)

- First order optimality conditions lead to Cone Complementarity Problem

Introduce the convex hypercone...
\[
\gamma = \left( \bigoplus_{i \in \mathcal{A}(q^l, \epsilon)} FC^i \right)
\]

\(\mathcal{F}^i \in \mathbb{R}^3\) represents friction cone associated with \(i^{th}\) contact

... and its dual hypercone:
\[
\gamma^* = \left( \bigoplus_{i \in \mathcal{A}(q^l, \epsilon)} FC^{i*} \right)
\]

CCP assumes following form: Find \(\gamma\) such that
\[
\gamma \in \gamma \quad \perp \quad N\gamma + d \in \gamma^*
\]
DVI: The QP Angle…

- The relaxed EOM represent a cone-complementarity problem (CCP)
- The CCP captures the first-order optimality condition for a cone-constrained quadratic optimization problem

\[
\begin{align*}
\min \quad & q(\gamma) = \frac{1}{2} \gamma^T N \gamma + d^T \gamma \\
\text{subject to} \quad & \gamma_i \in \Upsilon_i \text{ for } i = 1, 2, \ldots, N_c
\end{align*}
\]

- Notation used:

\[
\gamma \equiv [\gamma_1^T, \gamma_2^T, \ldots, \gamma_{N_c}^T]^T \in \mathbb{R}^{3 \times N_c} \quad \text{and} \quad \Upsilon_i : (\gamma_{u,i}^2 + \gamma_{w,i}^2) - \mu_i^2 \gamma_{n,i}^2 \leq 0
\]
Putting Things in Perspective…

- Problem solved at each time step: (advancing simulation from $t_l$ to $t_{l+1}$)
  \[
  \gamma \in \gamma \perp (N\gamma + d) \in \gamma^* \\
  v^{(l+1)} = M^{-1} \left( \tilde{k} + D\gamma \right) \\
  q^{(l+1)} = q^{(l)} + hL(q^{(l)})v^{(l+1)}
  \]

- Four key points led to above algorithm:
  - Coulomb Friction posed as an optimization problem
  - Working with velocity and impulses rather than acceleration and forces
  - Working with constraint equations (unilateral and bilateral) at the velocity level
  - Contact complementarity expression altered to lead to CCP
The Case for Parallel Computing

- Lots of contacts, handle them in parallel…
Implementation

- Method outlined implemented using two loops
  - Outer loop – runs the time stepping
  - Inner loop – CCP Algorithm (solves CCP problem at each time step)
Granular Dynamics: How Parallel Computing is Leveraged

1. Parallel Collision Detection
2. (Body parallel) Force kernel
3. (Contact parallel) Contact preprocessing kernel
4. (Contact parallel) CCP contact kernel
5. (Constraint parallel) CCP constraint kernel
6. (Reduction-slot parallel) Velocity reduction kernel
7. (Body parallel) Body velocity update kernel
11. (Body parallel) Time integration kernel
Inner Loop (CCP Algorithm)

1. For each contact $i$, evaluate $\eta_i = 3/\text{Trace}(D_i^T M^{-1} D_i)$.

2. If some initial guess $\gamma^*$ is available for multipliers, then set $\gamma^0 = \gamma^*$, otherwise $\gamma^0 = 0$.

3. Initialize velocities: $v^0 = \sum_i M^{-1} D_i \gamma_i^0 + M^{-1} \tilde{k}$.

4. For each contact $i$, compute changes in multipliers for contact constraints:

   $\gamma_i^{r+1} = \lambda \Pi_{\gamma_i} \left( \gamma_i^r - \omega \eta_i \left( D_i^T v^r + b_i \right) \right) + (1 - \lambda) \gamma_i^r$;

   $\Delta \gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$;

   $\Delta v_i = M^{-1} D_i \Delta \gamma_i^{r+1}$.

5. Apply updates to the velocity vector:

   $v^{r+1} = v^r + \sum_i \Delta v_i$

6. $r := r + 1$. Repeat from 4 until convergence or $r > r_{\text{max}}$.
Relying on Gauss-Seidel (if not running in parallel) or on Jacobi method

100% block-diagonal Jacobi implementation would look like:

$$\gamma^{(r+1)} = (I - D^{-1}N)\gamma^{(r)} - D^{-1}d$$

- Block diagonal matrix defined as $D = \text{diag}\{N_{11}, \ldots, N_{Nc,Nc}\}$

The software implements

$$\gamma^{(r+1)} = (I - \omega BN)\gamma^{(r)} - \omega Bd$$

- We use a block diagonal matrix $B = \text{diag}\{B_1, \ldots, B_{Nc}\}$ with
  $$B_i = \eta_i \frac{3}{\text{Trace}(N_{ii})} \cdot I_{3 \times 3} \in \mathbb{R}^{3 \times 3}$$
Example Application

- About 1.1 million bodies
- Objective function: about 15 million variables
- Simulated on the GPU
Experimental Setup

- Disruptor beads
- Nanopositioner controller
- CPU connection
- Load cell
- Translational stage
- Nanopositioner
Flow Measurement, 500 micron Spheres
Flow Simulation, 500 micron Spheres
Flow Measurement Results, 3mm Gap Size
Flow Measurement Results, 2.5mm Gap Size
Flow Measurement Results, 2mm Gap Size

[Graph showing weight over time for 2mm gap size]
Flow Measurement Results, 1.5mm Gap Size

![Graph showing weight versus time for different simulations with a gap size of 1.5mm](Image)
Validation Experiment: Repose Angle

- Experiment
- Simulation

\[ \alpha = 19.5^\circ \pm \] for \( t = 0.39 \)
Validation Experiment
Flow and Stagnation
Validation, Flow and Stagnation

![Graph showing weight in N over time for Gapsize 1.5mm with various lines representing different values and a red line indicating μ = 0.15.](image)
Validation, Flow and Stagnation

Gapsize 2.0mm

Weight in N

Time in s

μ = 0.15
Are We There Yet?

- Jacobi very slow to converge when moving into millions of bodies…
- One cubic meter of sand has 1.5 billion particles
- Currently seeking more robust and scalable approaches
GPMinRes Algorithm

- Initialize: $x_0$
- For $k=0$ to $N_{\text{max}}$
  - Let $y_0 = x_k$
  - Iterate over $j$ to compute $y_j$ using projected gradient descent. Accept $y_j$ when convergence slows down or the set of variables at their bounds is the same for consecutive iterates.
  - Update $x_k = y_j$.
  - Identify set of active constraints at $x_k$, compute selection matrix $Z_k$ for free set.
  - Solve with MINRES sub-problem $A_k w_k = -b_k$, where $A_k = Z_k^T N Z_k$ and $b_k = Z_k^T (N x_k + r)$
    - Note: this is unconstrained quadratic programming optimization problem
  - Use backtracking line-search with direction $d_k = Z_k w_k$ to compute $x_{k+1}$
  - If $\text{norm}(N x_{k+1} + r) < \text{tol}$ then break
- End
GPMinRes Parameters

- $N_{\text{max}}$: Max number of outer loops (100)
- $N_{\text{max,GP}}$: Max number of iterations in projected gradient descent step (20)
- $N_{\text{max,MINRES}}$: Max number of MINRES iterations to find solution of sub-problem (10)
- $N_{\text{max,PS}}$: Max number of iterations in projected search along direction $d_k$ (20)
- $\mu$: Sufficient decrease tolerance in projected search step (0.1)
- $\eta_1$: Decrease tolerance in projected gradient projection step (0.01)
- $\zeta$: Tolerance used to stop overall algorithm (1e-6)
Numerical Experiments

- Test: 1K, 2K, 4K bodies
  - For instance, 3525 contacts when 1K bodies
  - Keep the depth of the pile the same, increase the radius of the bucket…

- Contacts are frictionless

- Full $\mathbf{N}$ and $\mathbf{d}$ from C::E
GPMinRes with Preconditioning

- Recall that after determining the active set, the sub-problem $A_k w_k = -b_k$ is solved via MinRes
  - Comes from solving an unconstrained inner problem of smaller dimension

- Preconditioning: use a perturbed LU factorization when dealing with the inner MinRes problem
### Objective Function Value

[1K bodies, 3525 contacts]

<table>
<thead>
<tr>
<th>Method</th>
<th>Iterations</th>
<th>Final Objective Function Value</th>
<th>$\gamma_{\min}$</th>
<th>$\gamma_{\max}$</th>
<th>Computation Time [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPMINRES-no p</td>
<td>1000 MinRes Its. [within 100 changes of active set]</td>
<td>-2.9035</td>
<td>0.0</td>
<td>7.7487</td>
<td>6.7002</td>
</tr>
<tr>
<td>GPMINRES-no p (not plotted above)</td>
<td>10000 MinRes Its. [within 1000 changes of active set]</td>
<td>-2.9045</td>
<td>0.0</td>
<td>8.2002</td>
<td>61.0698</td>
</tr>
<tr>
<td>GPMINRES-p</td>
<td>100 MinRes Its. [within 100 changes of active set]</td>
<td>-2.8854</td>
<td>0.0</td>
<td>6.8551</td>
<td>1675</td>
</tr>
<tr>
<td>Jacobi</td>
<td>1000</td>
<td>-2.5077</td>
<td>0.0</td>
<td>4.4961</td>
<td>3.6643</td>
</tr>
</tbody>
</table>

The green & blue lines have 100 dots on them; i.e., 100 changes of active set.

The red line has 1000 dots on it; i.e., 1000 Jacobi sweeps.

The graph shows the objective function value over iterations for different methods and parameter configurations.
Size of Residual [1K, 2K, 4K Bodies]:
With and without preconditioning

Shown is the residual when solving the unconstrained internal optimization problem. It seems that not bothering to solve very accurately is not going to adversely impact the overall algorithm.
Size of Residual [1K, 2K, 4K Bodies]: With and without preconditioning

- Basically this is precisely the results of the previous slide reported in a different way.
- Each point on these plots represents an active set.
- The residual shown is the residual after the last MinRes iteration carried out to solve the optimization problem while on a face; i.e., keeping the same active set, solving an unconstrained problem.
Magnitudes of $x_k$ components

[1K bodies, 3525 contacts]

- Here, the solution vector $x_k$ is sorted by size and plotted.
- The blue dots represent the solution after 100 active sets (1,000 total MinRes iterations).
- The green dots represent the solution after 1000 active sets (10,000 total MinRes iterations).
- The red dots correspond to Jacobi after 1000 sweeps.
- The solution is ‘sharper’ when performing more iterations.
Magnitudes of $x_k$ components
[1K bodies, 3525 contacts]

This is basically the same data as the previous slide, this time plotted in histograms. Again, the results from 100 and 1000 active sets are quite similar.
History of Active Set Size

[1K bodies, 3525 contacts]

Plot shows the size of the active set each time the inner unconstrained sub-problem is solved.

- For some undetermined reason, the active set briefly becomes unsettled at about 850 active sets.
History of Active Set Size
[1K bodies, 3525 contacts]

- Note that the value of the objective function after 100 active sets reported a couple of slides back comes at a time when the active set is relatively unsettled.

- However, it is not drastically different than the value of the cost function after 1000 active set changes.
Magnitude of
Projected Gradient
[1K bodies, 3525 contacts]

Stopping criteria should be based on magnitude of projected gradient:

$$||\nabla_\Omega q(\gamma)|| \leq \tau$$

Projected gradient defined as

$$[\nabla_\Omega q(\gamma)]_i = \begin{cases} 
\partial_i q(\gamma) & \text{if } \gamma_i > 0 \\
\min(\partial_i q(\gamma), 0) & \text{if } \gamma_i = 0
\end{cases}$$
Magnitude of Contact Force
[upon exiting GPMinRes]
Magnitude of Contact Force
[upon exiting GPMinRes]
History of Active Set Size
Conclusions

- Jacobi is slow to reduce objective function
- The preconditioning is very expensive (at least in MATLAB) and doesn’t buy a whole lot
- GPMinRes without preconditioning looks peppy
- Going through many changes of active set doesn’t seem to buy much
  - Also, the objective function is rather insensitive to the size/content of the active set
- Performing a small number of inner iterations in MinRes when solving the unconstrained optimization problem at each step k doesn’t seem to be hurting
  - In other words, it’s ok to be sloppy and bail out when the residual is relatively large: 1E-3 in MinRes-noP compared to 1E-12 in MinRes-P
- Overall, compared to Jacobi, GPMinRes seems to be a step in the right direction
  - Decently straightforward to parallelize (has one global reduce per inner MinRes iteration though)
What comes next…

- Move to larger problems
  - On NVIDIA C2070 GPU we can test with problem having about 20 million unknowns

- It seems to be that although the objective function is not decreasing, when you keep iterating you get sharper values for $\theta$. Is this really the case?

- Move to friction case. Use as starting point Kucera SIOPT paper of 2008. There are some small issues with that paper, which is slightly sloppy at times.
Computational Dynamics: The Nuts and Bolts
Heterogeneous Cluster

Third fastest cluster at University of Wisconsin-Madison
Heterogeneous Computing Template (HCT): A Software Infrastructure for Large Scale Physics-Based Simulation

- Underlying theme of our lab’s effort
  - Develop a Heterogeneous Computing Template (HCT) to solve large engineering problems using new math on emerging hardware architectures

- Targeted “emerging hardware architectures”:
  - Clusters of CPUs and GPUs (accelerators)
    - More than 100 CPU cores, tens of GPU cards, tens of thousands of GPU cores
HCT: Five Major Components

- Computational Dynamics requires
  - Advanced modeling techniques
  - Strong algorithmic (applied math) support
  - Proximity computation
  - Domain decomposition & Inter-domain data exchange
  - Post-processing (visualization)

- HCT represents the library support, the associated API, and the embedded tools that support this five component abstraction
Computational Dynamics requires

- **Advanced modeling techniques**
- Strong algorithmic (applied math) support
- Proximity computation
- Domain decomposition & Inter-domain data exchange
- Post-processing (visualization)
HCT: Support for Advanced Modeling Techniques

- **Modeling (definition):**
  - The process of formulating a set of governing differential equations that captures the multi-physics associated with the engineering problem of interest

- **Issues:**
  - Modeling approaches are sometimes completely new or have seen little previous usage
  - Multi-physics: multiple spatial and temporal scales, difficult to solve

- **Worth investing time in modeling:** good modeling places you at an advantage when it comes to simulating hard problems
Multi-Body Dynamics

- Kinematic Differential Equations
- Force Balance Equations
- Holonomic Kinematic Constraints

Generalized Positions

Generalized Mass Matrix

Velocity Transformation Matrix

Generalized Velocities

\[ \dot{q} = L(q)\dot{v} \]

\[ M(q)\ddot{v} = f(t, q, v) - g_q^T(q, t)\lambda \]

\[ g(q, t) = 0 \]

Reaction Force

Applied Force
Many-Body Dynamics

- Generalized Positions
- Generalized Mass Matrix
- Kinematic Differential Equations
- Force Balance Equations
- Holonomic Kinematic Constraints
- Contact Complementarity Conditions
- Coulomb Friction Model

\[ \dot{q} = L(q) \dot{v} \]
\[ M(q) \dot{v} = \mathbf{f}(t, q, v) - g^T_q(q, t) \lambda + \sum_{i=1}^{N_c} (\gamma_n^i D_n^T i + \gamma_u^i D_u^T i + \gamma_w^i D_w^T i) \]
\[ g(q, t) = 0 \]

- Velocity Transformation Matrix
- Generalized Velocities
- Reaction Force
- Frictional Contact Force

Contact Impulse, for Contact “i”
\[ 0 \leq \Phi^i(q, t) \quad \perp \quad \gamma_n^i \geq 0 \quad \text{for } i = 1, 2, \ldots, N_c \]

Friction Impulse Components, for Contact “i”
\[ (\gamma_n^i, \gamma_w^i) = \arg \min_{\gamma_n^i, \gamma_w^i \geq 0} \frac{\sqrt{(\gamma_n^i)^2 + (\gamma_w^i)^2}}{\sqrt{\gamma_n^i}} \]

Gap Function, for Contact “i”
\[ \frac{\gamma_n^T D_n^i + \gamma_u^T D_u^i + \gamma_w^T D_w^i}{\text{Total Number of Contacts}} \]

Friction Dissipation Energy
Fluid-Solid Interaction Example

- Separating living/dead cells
Multi-Physics: Multi-Body Dynamics & Fluid Dynamics

Kinematic Differential Equations

Force Balance Equations

Holonomic Kinematic Constraints

Generalized Positions

Generalized Mass Matrix

Velocity Transformation Matrix

Generalized Velocities

Reaction Force

Applied Force

Conservation of mass:
\[
\frac{d\rho}{dt} = -\rho \frac{\partial v^\beta}{\partial x^\beta} \Rightarrow \frac{d\rho_i}{dt} \approx \sum_j m_j v_{ij} \cdot \nabla_i W_{ij}
\]

Conservation of momentum:
\[
\frac{d\sigma^\alpha}{dt} = \frac{1}{\rho} \frac{\partial \sigma^\alpha}{\partial x^\beta} + \frac{f^\alpha}{\rho} \Rightarrow \frac{d \sigma_{ij}}{dt} \approx -\sum_j m_j \left( \frac{p_{ij}}{\rho_j^2} + \frac{p_i}{\rho_i^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \frac{f_{ij}}{m_i}
\]

Conservation of energy:
\[
\frac{du}{dt} = \frac{\sigma^\alpha}{\rho} \frac{\partial v^\alpha}{\partial x^\beta} \Rightarrow \frac{d u_i}{dt} \approx \frac{1}{2} \sum_j m_j \left( \frac{p_{ij}}{\rho_j^2} + \frac{p_i}{\rho_i^2} + \Pi_{ij} \right) v_{ij} \cdot \nabla_i W_{ij}
\]
Simulation results: velocity magnitude
Simulation results
Computational Dynamics requires

- Advanced modeling techniques
- Strong algorithmic (applied math) support
- Proximity computation
- Domain decomposition & Inter-domain data exchange
- Post-processing (visualization)
CCP Solution Methods Tested

- MATLAB native algorithms (unconstrained approaches)
  - Preconditioned conjugate gradient (PCG)
  - Generalized minimum residual (GMRES)
  - Minimum residual (MINRES)

- Implemented by us: frictionless case (bound constraints in place)
  - Jacobi (CE)
  - Projected conjugate gradient (ProjCG)
  - Gradient projected conjugate gradient (GPCG)
  - Gradient projection MinRes (GPMinRes)

- Implemented by us: friction case (cone constraints - ongoing)
  - Active-set algorithm for minimizing quadratic functions with separable convex constraints (Kučera, SIOPT 2008)
Computational Dynamics requires

- Advanced modeling techniques
- Strong algorithmic (applied math) support
- **Proximity computation**
- Domain decomposition & Inter-domain data exchange
- Post-processing (visualization)
CD: Binning

- Example: 2D collision detection, bins are squares

- Body 4 touches bins A4, A5, B4, B5
- Body 7 touches bins A3, A4, A5, B3, B4, B5, C3, C4, C5
- In proposed algorithm, bodies 4 and 7 will be checked for collision by three threads (associated with bin A4, A5, B4)
Stage 1 (Body Parallel)

- Purpose: find the number of bins touched by each body
- Store results in the “T”, array of N integers
- Key observation: it’s easy to bin bodies
Stage 2: Parallel Inclusive Scan

- Run a parallel inclusive scan on the array $T$
  - The last element is the total number of bin touches, including the last body

- Complexity of Stage: $O(N)$ – thrust library

- Purpose: determine the number of entries $M$ needed to store the indices of all the bins touched by each body in the problem
Stage 3: Determine bin-to-body association

- Stage executed in parallel on a per-body basis

- Allocate an array \( B \) of \( M \) pairs of integers.
  - The key (first entry of the pair), is the bin index
  - The value (second entry of pair) is the body that touches that bin
Stage 4: Radix Sort

- In parallel, run radix sort to order the B array according to key values

- Work load: $O(N)$
  - Relies on thrust library
Stage 5: Find Bin Starting Index

- Host allocates on device an array of length $N_b$ of pairs of unsigned integers

- Run in parallel, on a per bin basis:
  - Load in parallel in shared memory chunks of the $B$ array and find the location where each bin starts
  - Store it in entry $k$ of $C$, as the key associated with this pair
  - Key of bins with one or no bodies is set to maximum unsigned int value of 0xffffffff
Stage 6: Sort C for Pruning

- Do a parallel radix sort on the array C based on the key
- Purpose: move unused bins to the end of array
- Effort: $O(N_b)$
Stage 7: Investigate Collisions in each Bin

- Carried out in parallel, one thread per bin

To store information generated during this stage, host needs to allocate an unsigned integer array $D$ of length $N_b$
- Array $D$ stores the number of actual contacts occurring in each bin
- $D$ is in sync with (linked to) $C$, which in turn is in sync with (linked to) $B$

Parallelism: one thread per bin
- Thread $k$ reads the pair key-value in entry $k$ of array $C$
- Thread $k$ reads does rehearsal for brute force collision detection
- Outcome: the number $s$ of active collisions taking place in a bin
  - Value $s$ stored in $k^{th}$ entry of the $D$ array
Stage 7, details...

- In order to carry out this stage you need to keep in mind how C is organized, which is a reflection of how B is organized.

The drill: thread 0 relies on info at C[0], thread 1 relies on info at C[1], etc.

Let’s see what thread 2 (goes with C[2]) does:
- Read the first 2 bodies that start at offset 6 in B.
- These bodies are 4 and 7, and as B indicates, they touch bin A4.
- Bodies 4 and 7 turn out to have 1 contact in A4, which means that entry 2 of D needs to reflect this.
Stage 7, details

- Brute Force CD rehearsal
  - Carried out to understand the memory requirements associated with collisions in each bin
    - Finds out the total number of contacts owned by a bin
  - Key question: which bin does a contact belong to?
    - Answer: It belongs to bin containing the CM of the Contact Volume (CMCV)
Stage 7, Comments

- Two bodies can have multiple contacts, handled ok by the method

- Easy to define the CMCV for two spheres, two ellipsoids, and a couple of other simple geometries

  - In general finding CMCV might be tricky
    - Notice picture below, CM of 4 is in A5, CM of 7 is in B4 and CMCV is in A4

  - Finding the CMCV is the subject of the so called “narrow phase collision detection”
    - It’ll be simple in our case since we are going to work with simple geometry primitives
Stage 8: Inclusive Prefix Scan

- Save to the side the number of contacts in the last bin (last entry of $D$) $d_{\text{last}}$
  - Last entry of $D$ will get overwritten

```
0   1   2   3   4   ...
0   1   0   0   0   ...
(A2) (A4) (A5) (B1) (A3) ...
```

- Run parallel exclusive prefix scan on $D$:

```
0   1   2   3   4   ...
0   0   1   1   1   ...
(A2) (A4) (A5) (B1) (A3) ...
```

- Total number of actual collisions:

$$N_c = D[N_b] + d_{\text{last}}$$
Stage 9: Populate Array E

- From the host, allocate on the device memory for array E
  - Array E stores the required collision information: normal, two tangents, etc.
  - Number of entries in the array: $N_c$ (see previous slide)

- In parallel, on a per bin basis (one thread/bin):
  - Populate the E array with required info

- Not discussed in greater detail, this is just like Stage 7, but now you have to generate actual collision info (stage 7 was the rehearsal)

- Thread for A4 will generate the info for contact “c”
- Thread for C2 will generate the info for “i” and “d”
- Etc.
Stage 9, details

- B, C, D required to populate array E with collision information

- C and B are needed to compute the collision information

- D is needed to understand where the collision information will be stored in E
Ellipsoid-Ellipsoid CD: Visualization
Example: Ellipsoid-Ellipsoid CD

\[
d = P_1 - P_2 = \left( \frac{1}{2\lambda_1} M_1 + \frac{1}{2\lambda_2} M_2 \right)c + (b_1 - b_2)
\]

\[
\frac{\partial d}{\partial \alpha_i} = \frac{\partial P_1}{\partial \alpha_i} - \frac{\partial P_2}{\partial \alpha_i}, \quad \frac{\partial^2 d}{\partial \alpha_i \partial \alpha_j} = \frac{\partial^2 P_1}{\partial \alpha_i \partial \alpha_j} - \frac{\partial^2 P_2}{\partial \alpha_i \partial \alpha_j}
\]

\[
\frac{\partial P}{\partial \alpha_i} = \left( \frac{1}{2\lambda} M - \frac{1}{8\lambda^3} \text{Mcc}^T \text{M} \right) \frac{\partial c}{\partial \alpha_i}
\]

\[
\frac{\partial^2 P}{\partial \alpha_i \partial \alpha_j} = \left( -\frac{1}{8\lambda^3} M + \frac{3}{32\lambda^5} \text{Mcc}^T \text{M} \text{c}^T \text{M} \frac{\partial c}{\partial \alpha_j} \frac{\partial c}{\partial \alpha_i} \right) - \frac{1}{8\lambda^3} \left[ \left( \text{c}^T \text{M} \frac{\partial c}{\partial \alpha_i} \right) \text{M} + \text{M} \left( \frac{\partial c}{\partial \alpha_i} \right)^T \text{M} \right] \frac{\partial c}{\partial \alpha_j}
\]

\[
\frac{1}{2\lambda} M - \frac{1}{8\lambda^3} \text{Mcc}^T \text{M} \frac{\partial^2 c}{\partial \alpha_i \partial \alpha_j}
\]

\[
\varepsilon : \frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} + \frac{z^2}{r_3^2} = 1
\]

\[\Lambda : \text{Rotation Matrix}
\]

\[\text{M} = \Lambda R^2 \Lambda^T
\]

\[\text{R} = \text{diag}(r_1, r_2, r_3)
\]

\[\text{b} : \text{Translation of ellipsoids center}
\]

\[\lambda^2 = \frac{1}{4} n^T \text{M} n
\]

\[d = P_1 - P_2
\]

\[
\min_{\alpha_1, \alpha_2} \|d(\alpha_1, \alpha_2)\|^2
\]
Ellipsoid-Ellipsoid CD: Results

**Time vs. Number of Contacts**

- Time (seconds) vs. Number of Contacts
- Data points indicate a linear relationship between time and number of contacts.

**Speedup - GPU vs. CPU**

- Speedup vs. Number of Contacts
- Graph shows a rapid increase in speedup with increasing number of contacts.

11/21/11
Simulation Based Engineering Lab
Speedup - GPU vs. CPU (Bullet library)
[results reported are for spheres]

GPU: NVIDIA Tesla C1060
CPU: AMD Phenom II Black X4 940 (3.0 GHz)
Parallel Implementation: Number of Contacts vs. Detection Time [results reported are for spheres]
Multiple-GPU Collision Detection

Assembled Quad GPU Machine

Processor: AMD Phenom II X4 940 Black
Memory: 16GB DDR2
Graphics: 4x NVIDIA Tesla C1060
Power supply 1: 1000W
Power supply 2: 750W
SW/HW Setup

Open MP

CUDA

Main Data Set

Results

Thread 0
Thread 1
Thread 2
Thread 3

GPU 0
GPU 1
GPU 2
GPU 3

16 GB RAM

Quad Core AMD Microprocessor

Tesla C1060 4x4 GB Memory 4x30720 threads

11/21/11
Simulation Based Engineering Lab
Results – Contacts vs. Time

Quad Tesla C1060 Configuration

Time (Sec)

Contacts (Billions)
Computational Dynamics requires

- Advanced modeling techniques
- Strong algorithmic (applied math) support
- Proximity computation
- **Domain decomposition & Inter-domain data exchange**
- Post-processing (visualization)
Computation Using Multiple CPUs
HCT: Domain decomposition & Inter-domain data exchange

- Relates to the ability to divide the simulation into chunks and have multiple CPUs/GPUs exchange data during simulation as needed.

- Elements leave one subdomain to move to a different one.

- Key issues:
  - Dynamic load balancing
  - Establish a dynamic data exchange protocol (DDEP) between sub-domains.
Computation Using Multiple CPUs
Computation Using Multiple CPUs
Computational Dynamics requires

- Advanced modeling techniques
- Strong algorithmic (applied math) support
- Proximity computation
- Domain decomposition & Inter-domain data exchange
- **Post-processing (visualization)**
HCT: Visualization and Post-Processing

- Rendering very complex scenes with more than one million components
- Rendering takes longer than simulating
- Pursuing a rendering pipeline that draws on multiple CPUs and GPUs
Track Simulation 1

Parameters:

- Driving speed: 1.0 rad/sec
- Length: 12 seconds
- Time step: 0.005 sec
- Computation time: 18.5 hours
- Particle radius: .027273 m
- Terrain: 284,715 particles
- Inertia parameters of track are fake
Track Simulation 2

Parameters:

• Driving speed: 1.0 rad/sec
• Length: 10 seconds
• Time step: 0.005 sec
• Computation time: 17.8 hours
• Particle radius: 0.025±0.0025 m
• Terrain: 467,100 particles
• Inertia parameters of track are fake
Dual Track ‘Footprint’
Simulation of MRAP Impacted by Debris
Simulation of MRAP Impacted by Debris

Animations show work in progress
Simulation of MRAP Impacted by Debris
HCT: Five Major Components
Looking Ahead...

- Novel modeling techniques
  - Rigid/Deformable bodies, fluid-solid interaction, electrostatics, cohesion

- Strong algorithmic (applied math) support
  - Focus on algorithms suitable for parallel computing

- Proximity computation
  - Handling complex non-convex topologies + continuous contact detection algorithm

- Domain decomposition & Inter-domain data exchange
  - Load balancing in distributed heterogeneous computing

- Post-processing (visualization)
  - Establish a feature-rich ready-to-use rendering pipeline
HCT at Work

- HCT is being built through and used in several projects
  - US Army TARDEC – vehicle mobility on deformable terrain
  - NASA’s JPL – anchoring in granular materials
  - Caterpillar – terramechanics and collision detection projects
  - Dept. of Biomechanics at UW-Madison - FSI
  - NSF - Deformable body mechanics and large scale DEM
Conclusions

- Developing a Heterogeneous Computing Template (HCT) to solve large engineering problems on emerging hardware architectures

- Targeted “emerging hardware architectures”:  
  - Clusters of CPUs and GPUs (accelerators)  
    - More than 100 CPU cores, tens of GPU cards, tens of thousands of GPU cores
Thank You.
negrut@wisc.edu